IN THE CLAIMS

Amended claims follow:

1. (Currently Amended) A method for shadow mapping~~ while rendering a primitive in a graphics pipeline~~, comprising:
~~(a)~~ performing an offset operation to generate a depth value~~ while rendering a primitive~~;
~~(b)~~ identifying a value of a slope~~ associated with a primitive~~; and
~~(c)~~ conditionally clamping the depth value based on the value of the slope.

2. (Currently Amended) The method as recited in claim 1, wherein the shadow mapping process includes rendering ~~the~~a primitive from a light space perspective.

3. (Original) The method as recited in claim 1, wherein the depth value is clamped if the value of the slope is greater than a predetermined amount.

4. (Currently Amended) The method as recited in claim 1, wherein the clamping includes the steps of: identifying vertex depth values of vertices of ~~the~~a primitive; comparing at least one of the vertex depth values with the depth value generated by the offset operation; and clamping the depth value generated by the offset operation based on the comparison.

5. (Original) The method as recited in claim 4, wherein the depth value generated by the offset operation is clamped to the depth value generated by the offset operation if the depth value generated by the offset operation is less than the maximum vertex depth value.

6. (Original) The method as recited in claim 4, wherein the depth value generated by the offset operation is clamped to the greatest vertex depth

value if the greatest vertex depth value is less than the depth value generated by the offset operation.

7. (Original) The method as recited in claim 4, wherein the depth value generated by the offset operation is clamped to the depth value generated by the offset operation if the depth value generated by the offset operation is greater than the least one of the vertex depth values.

8. (Original) The method as recited in claim 4, wherein the depth value generated by the offset operation is clamped to the least one of the vertex depth values if the least one of the vertex depth values is greater than the depth value generated by the offset operation.

9. (Original) The method as recited in claim 1, wherein the offset operation includes a polygon offset operation in accordance with the OpenGL® programming language.

10. (Currently Amended) A computer program embodied on a computer readable medium for shadow mapping~~while rendering a primitive in a graphics pipeline~~, comprising:

~~(a)~~——a code segment for performing an offset operation to generate a depth value ~~while rendering a primitive~~;

~~(b)~~——a code segment for identifying a value of a slope~~associated with a slope of the primitive~~; and

~~(c)~~——a code segment for conditionally clamping the depth value based on the value of the slope.

11. (Original) The computer program as recited in claim 10, wherein the depth value is clamped if the value of the slope is greater than a predetermined amount.

12. (Currently Amended) The computer program as recited in claim 10, wherein the clamping includes: identifying vertex depth values of vertices of ~~the~~a

primitive; comparing at least one of the vertex depth values with the depth value generated by the offset operation; and clamping the depth value generated by the offset operation based on the comparison.

13. (Original) The computer program as recited in claim 12, wherein the depth value generated by the offset operation is clamped to the depth value generated by the offset operation if the depth value generated by the offset operation is less than the maximum vertex depth value, and wherein the depth value generated by the offset operation is clamped to the greatest vertex depth value if the greatest vertex depth value is less than the depth value generated by the offset operation.

14. (Original) The computer program as recited in claim 12, wherein the depth value generated by the offset operation is clamped to the depth value generated by the offset operation if the depth value generated by the offset operation is greater than the least one of the vertex depth values, and wherein the depth value generated by the offset operation is clamped to the least one of the vertex depth values if the least one of the vertex depth values is greater than the depth value generated by the offset operation.

15. (Currently Amended) A system for shadow mapping ~~while rendering a primitive in a graphics pipeline~~, comprising:
~~(a)~~ logic for performing an offset operation to generate a depth value ~~while rendering a primitive~~;
~~(b)~~ logic for calculating and identifying a value of a slope ~~associated with the primitive~~; and
~~(c)~~ logic for conditionally clamping the depth value based on the value of the slope.~~;~~

16. – 28. (Cancelled)

29. (New) A method for performing shading calculations in a graphics pipeline, comprising:

performing a first shading calculation in order to generate output utilizing a graphics pipeline;
saving the output; and
performing a second shading calculation using the output in order to generate further output utilizing the graphics pipeline.

30. (New) The method as recited in claim 29, wherein the first shading calculation includes [(1-s)*(Color_diff + Color_spec)] for generating an output A, and the second shading calculation includes [Color_amb + A], where s is a shadow variable, Color_diff is a diffuse color variable, Color_spec is a specular color variable, and Color_amb is an ambient color variable.

31. (New) The method as recited in claim 29, wherein the first shading calculation includes [((1-s)* Color_diff) + Color_amb] for generating an output A, and the second shading calculation includes [A*Texture_det + (1-s)* Color_spec], where s is a shadow variable, Color_diff is a diffuse color variable, Color_spec is a specular color variable, Color_amb is an ambient color variable, and Texture_det is a detail texture variable.

32. (New) The method as recited in claim 29, wherein the first and second shading calculations together include a diffuse color variable, a specular color variable, and an ambient color variable.

33. (New) The method as recited in claim 32, wherein the variables are decoupled.

34. (New) The method as recited in claim 29, wherein the method is carried out with a system comprising:
(a) a shading module for performing the first shading calculation in order to generate the output;
(b) a texture look-up module coupled to the shading module for retrieving texture information using texture coordinates associated with the output;

(c) a feedback loop coupled between an input and an output of the shading module for performing the second shading calculation using the texture information from the texture look-up module in order to generate further output; and

(d) a combiner module coupled to the output of the shading module for combining the output generated by the shading module.

35. (New) A computer program embodied on a computer readable medium for performing shading calculations in a graphics pipeline, comprising:
a code segment for performing a first shading calculation in order to generate output utilizing a graphics pipeline;
a code segment for saving the output; and
a code segment for performing a second shading calculation using the output in order to generate further output utilizing the graphics pipeline.

36. (New) The computer program as recited in claim 35, wherein the first shading calculation includes [(1-s)*(Color_diff + Color_spec)] for generating an output A, and the second shading calculation includes [Color_amb + A], where s is a shadow variable, Color_diff is a diffuse color variable, Color_spec is a specular color variable, and Color_amb is an ambient color variable.

37. (New) The computer program as recited in claim 35, wherein the first shading calculation includes [((1-s)* Color_diff) + Color_amb] for generating an output A, and the second shading calculation includes [A*Texture_det + (1-s)* Color_spec], where s is a shadow variable, Color_diff is a diffuse color variable, Color_spec is a specular color variable, Color_amb is an ambient color variable, and Texture_det is a texture-detail variable.

38. (New) The computer program as recited in claim 35, wherein the first and second shading calculations together include a diffuse color variable, a specular color variable, and an ambient color variable.

39. (New) The computer program as recited in claim 38, wherein the variables are decoupled.

40. (New) The computer program as recited in claim 35, wherein the code segments are carried out with a system comprising:

(a) a shading module for performing the first shading calculation in order to generate the output;

(b) a texture look-up module coupled to the shading module for retrieving texture information using texture coordinates associated with the output;

(c) a feedback loop coupled between an input and an output of the shading module for performing the second shading calculation using the texture information from the texture look-up module in order to generate further output; and

(d) a combiner module coupled to the output of the shading module for combining the output generated by the shading module.

41. (New) A system for performing shading calculations in a graphics pipeline, comprising:

logic for performing a first shading calculation in order to generate output utilizing a graphics pipeline;

logic for saving the output; and

logic for performing a second shading calculation using the output in order to generate further output utilizing the graphics pipeline.

42. (New) The method as recited in claim 29, wherein the shading calculations involve shadow modulation.

43. (New) The method as recited in claim 42, wherein the shadow modulation involves more than one function.

44. (New) The method as recited in claim 29, wherein the first and second shading calculations involve decoupled variables.